Crypto for Java Developers

Java Forum Stuttgart 2014

Dominik Schadow - bridgingIT

„Crypto is complex and full of backdoors…“

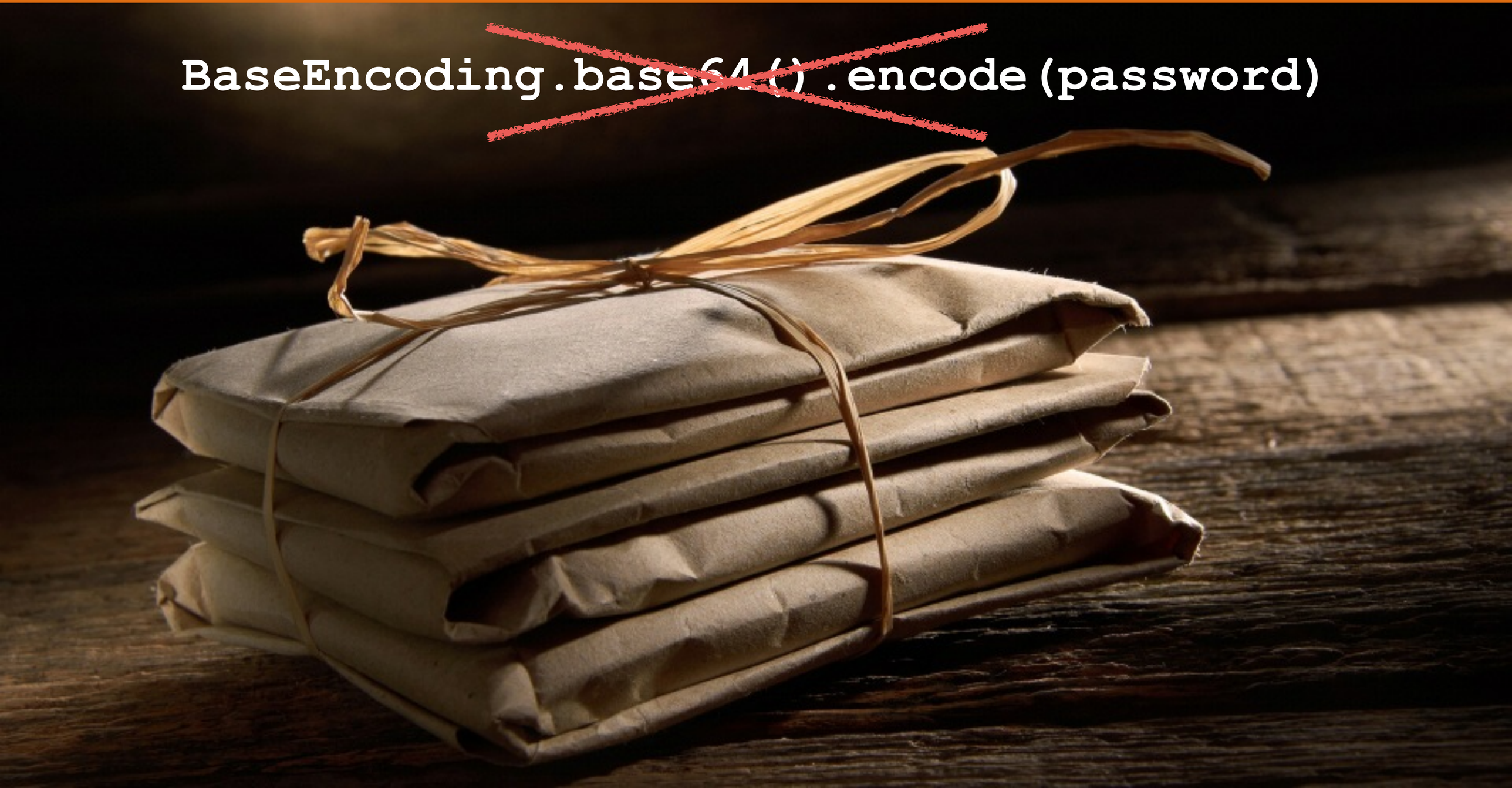
A6 Sensitive Data Exposure

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability DIFFICULT	Prevalence UNCOMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider who can gain access to your sensitive data and any backups of that data. This includes the data at rest, in transit, and even in your customers' browsers. Include both external and internal threats.	Attackers typically don't break crypto directly. They break something else, such as steal keys, do man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's browser.	The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm usage is common, particularly weak password hashing techniques. Browser weaknesses are very common and easy to detect, but hard to exploit on a large scale. External attackers have difficulty detecting server side flaws due to limited access and they are also usually hard to exploit.		Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive data such as health records, credentials, personal data, credit cards, etc.	Consider the business value of the lost data and impact to your reputation. What is your legal liability if this data is exposed? Also consider the damage to your reputation.

There is a pattern in most data breaches…

1. All passwords are safe!
2. Stolen, but securely hashed!
3. OK, some have a simple hash…
4. Change your password now!

No secret, no key -> no security




security
attack
random
keystore
hashes
plaintext
signatures
decryption
secret
management
passwords
data
digital
keys
id
asymmetric
private
authentication
dictionary
encryption
exchange
certificates
brute-force
verification
sensitive
transport
public
configuration
pair
numbers
hybrid
session
symmetric
information
ciphertext

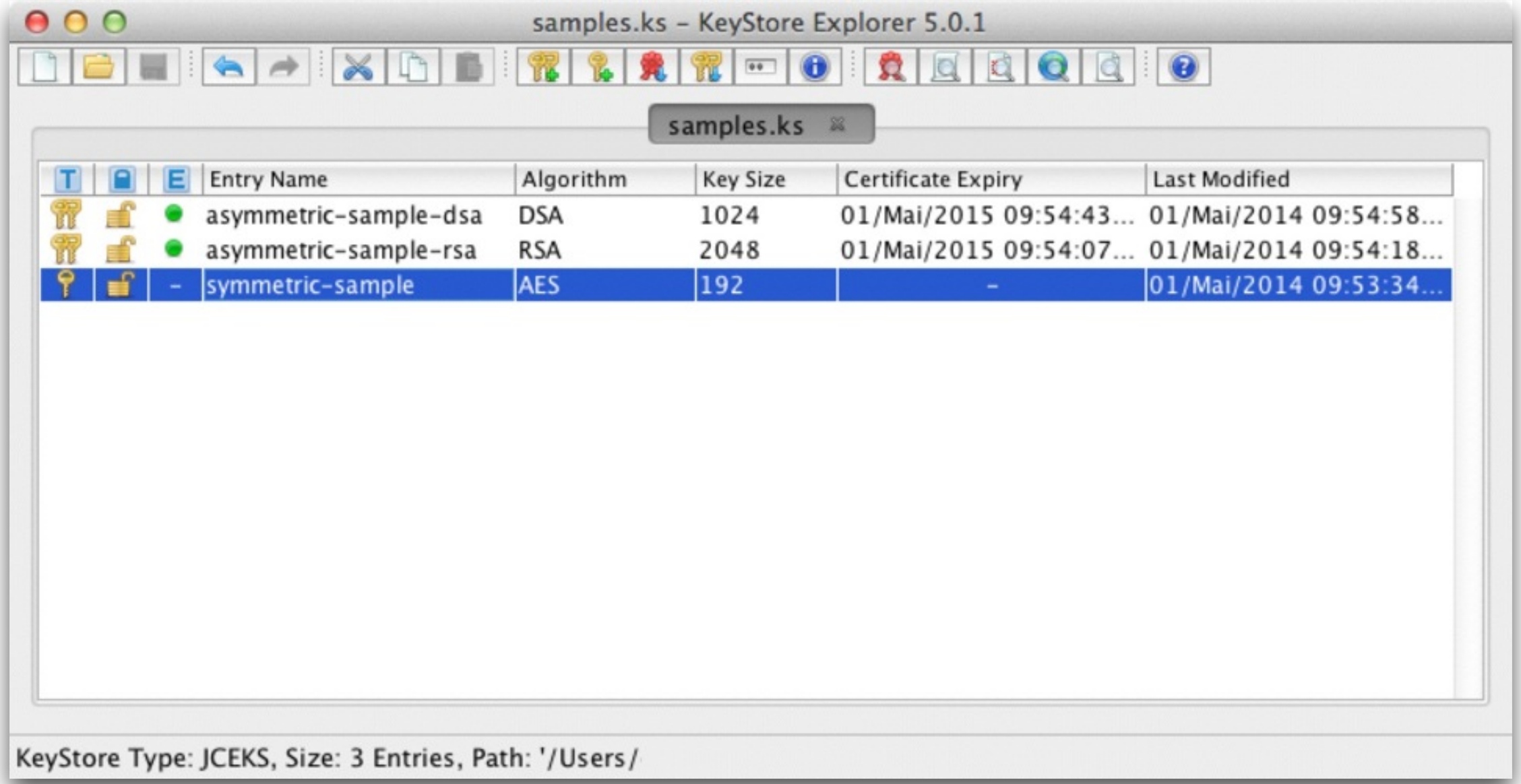
Pure Java
Spring Configuration
Encryption & Signatures
Hashes

Pure Java

Java KeyStore stores key pairs, secret keys and certificates

- Common types: PK12, **JCEKS**
- Store passwords since Java 8
- KeyTool on command line

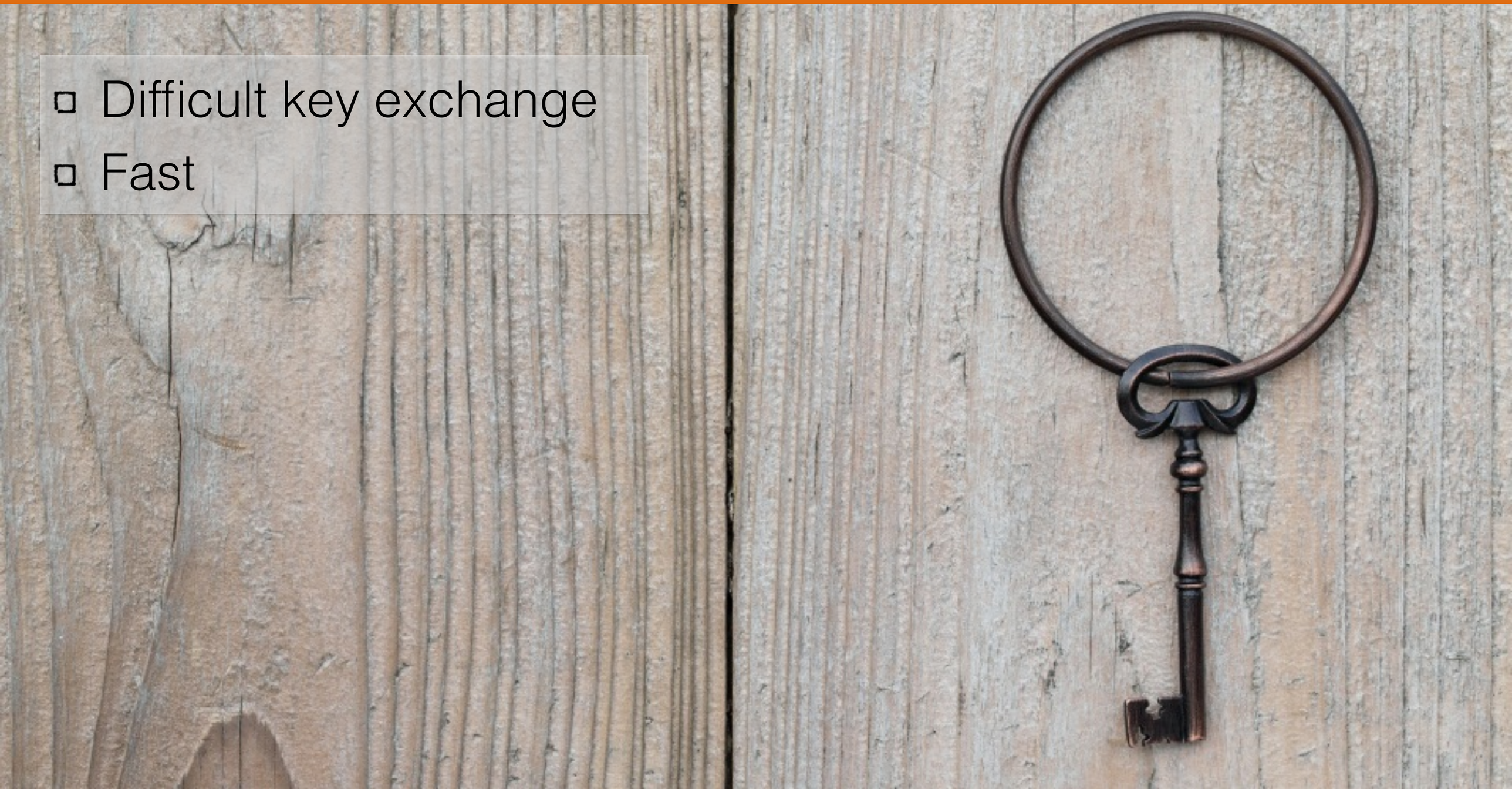
Easy keystore management with KeyStore Explorer



keystore-explorer.sourceforge.net

One shared secret key for symmetric encryption

- Difficult key exchange
- Fast



Individual public and private key for asymmetric encryption

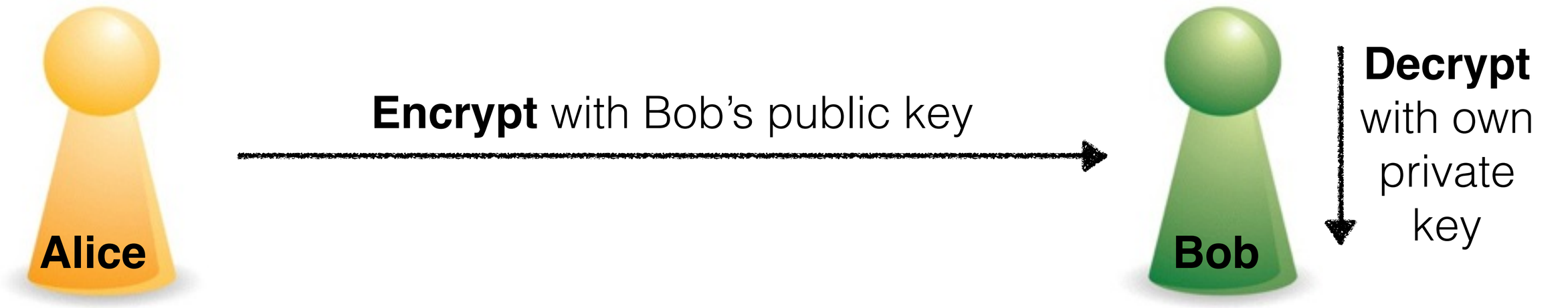
- Simple key exchange
- Slower

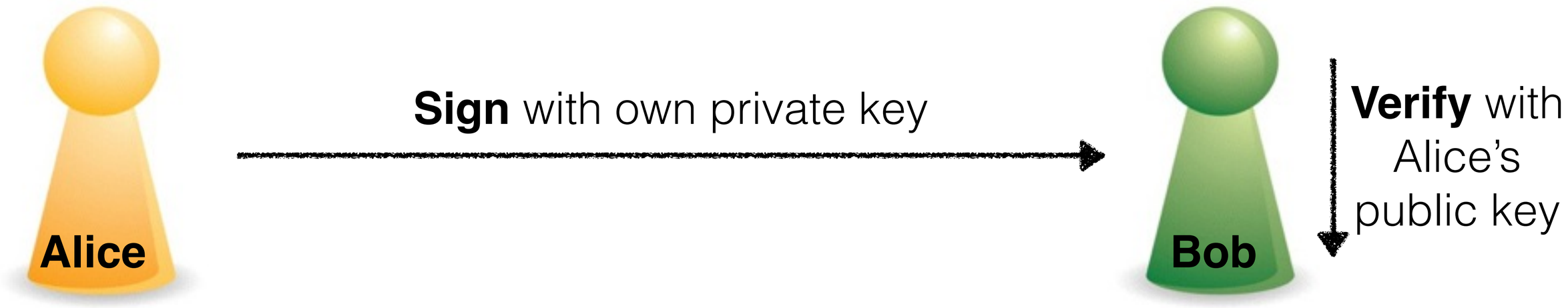
Hybrid encryption

Content: Symmetric

Key: Asymmetric

Encrypt with counterparts public key, sign with own private key





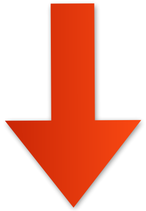
Demo

(Crypto) frameworks make crypto easier, but come with limitations

Spring Configuration

Typical DataSource configuration with Spring

```
<context:property-placeholder
  location="classpath:spring.properties" />

<bean id="dataSource" class="org...BasicDataSource">
  <!-- … -->
  <property name="url" value="${jdbc.url}"/>
  <property name="username" value="${jdbc.un}"/>
  <property name="password" value="${jdbc.pw}"/>
</bean>
```



```
jdbc.driver=org.hsqldb.jdbcDriver
jdbc.url=jdbc:hsqldb:mem:sampleDB
jdbc.un=sampleUser
jdbc.pw=samplePassword
```

Jasypt command line tools for password generation

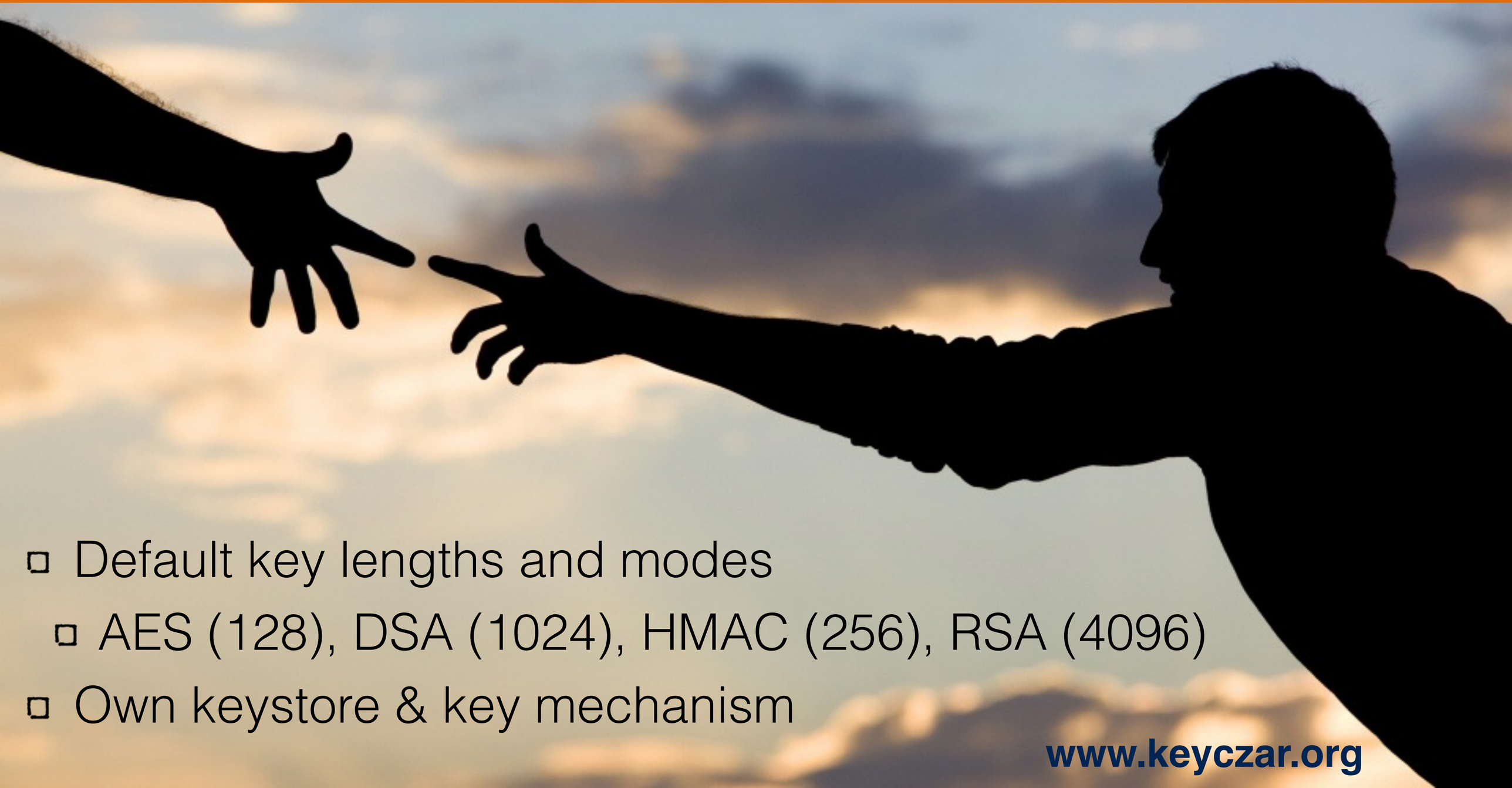
bin — bash — 120×32

```
marvin:bin dos$ ./listAlgorithms.sh

DIGEST ALGORITHMS:   [MD2, MD5, SHA, SHA-224, SHA-256, SHA-384, SHA-512]

PBE ALGORITHMS:      [PBEWITHHMACSHA1ANDAES_128, PBEWITHHMACSHA1ANDAES_256, PBEWITHHMACSHA224ANDAES_128, PBEWITHHMACSHA2
24ANDAES_256, PBEWITHHMACSHA256ANDAES_128, PBEWITHHMACSHA256ANDAES_256, PBEWITHHMACSHA384ANDAES_128, PBEWITHHMACSHA384AN
DAES_256, PBEWITHHMACSHA512ANDAES_128, PBEWITHHMACSHA512ANDAES_256, PBEWITHMD5ANDDES, PBEWITHMD5ANDTRIPLEDES, PBEWITHSHA
1ANDDESEDE, PBEWITHSHA1ANDRC2_128, PBEWITHSHA1ANDRC2_40, PBEWITHSHA1ANDRC4_128, PBEWITHSHA1ANDRC4_40]

marvin:bin dos$ ./encrypt.sh input="samplePassword" password="spring-jasypt" algorithm="PBEWITHSHA1ANDDESEDE"

----ENVIRONMENT----------------

Runtime: Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 25.5-b02



----ARGUMENTS------------------

algorithm: PBEWITHSHA1ANDDESEDE
input: samplePassword
password: spring-jasypt



----OUTPUT----------------------

s6eA5w9OtgVM0vZ1hjKH/w1ndwJiGMr0


marvin:bin dos$
```

Demo

Encryption & Signatures

Keyczar offers safe algorithms for encrypting and signing



- Default key lengths and modes
  - AES (128), DSA (1024), HMAC (256), RSA (4096)
- Own keystore & key mechanism

www.keyczar.org

Keyczar uses JSON format for key sets and keys

KeyczarTool: java -jar KeyczarTool-0.71g-090613.jar

symmetric — bash — 120×36

```
Usage: "KeyczarTool command flags"
Commands: create addkey pubkey promote demote revoke usekey importkey exportkey
Flags: location name size status purpose padding destination version asymmetric crypter pemfile passphrase
Command Usage:
create --location=/path/to/keys --purpose=(crypt|sign) [--name="A name"] [--asymmetric=(dsa|rsa|ec)]
        Creates a new, empty key set in the given location.
        This key set must have a purpose of either "crypt" or "sign"
        and may optionally be given a name. The optional version
        flag will generate a public key set of the given algorithm.
        The "dsa" and "ec" asymmetric values are valid only for sets
        with "sign" purpose.

addkey --location=/path/to/keys [--status=(active|primary)] [--size=size] [--crypter=crypterLocation] [--padding=(OAEP|P
KCS)]
        Adds a new key to an existing key set. Optionally
        specify a status, which is active by default. Optionally
        specify a key size in bits. Also optionally specify the
        location of a set of crypting keys, which will be used to
        encrypt this key set.  The optional --padding flag is allowed
        only for key sets created with --version=rsa.  If omitted, it
        defaults to OAEP.
pubkey --location=/path/to/keys --destination=/destination
        Extracts public keys from a given key set and writes them
        to the destination. The "pubkey" command Only works for
        key sets that were created with the "--asymmetric" flag.
promote --location=/path/to/keys --version=versionNumber
        Promotes the status of the given key version in the given
        location. Active keys are promoted to primary (which demotes
        any existing primary key to active). Inactive keys are
        promoted to be active.
demote --location=/path/to/keys --version=versionNumber
        Demotes the status of the given key version in the given
        location. Primary keys are demoted to active. Active keys
        are made inactive.
revoke --location=/path/to/keys --version=versionNumber
        Revokes the key of the given version number.
```

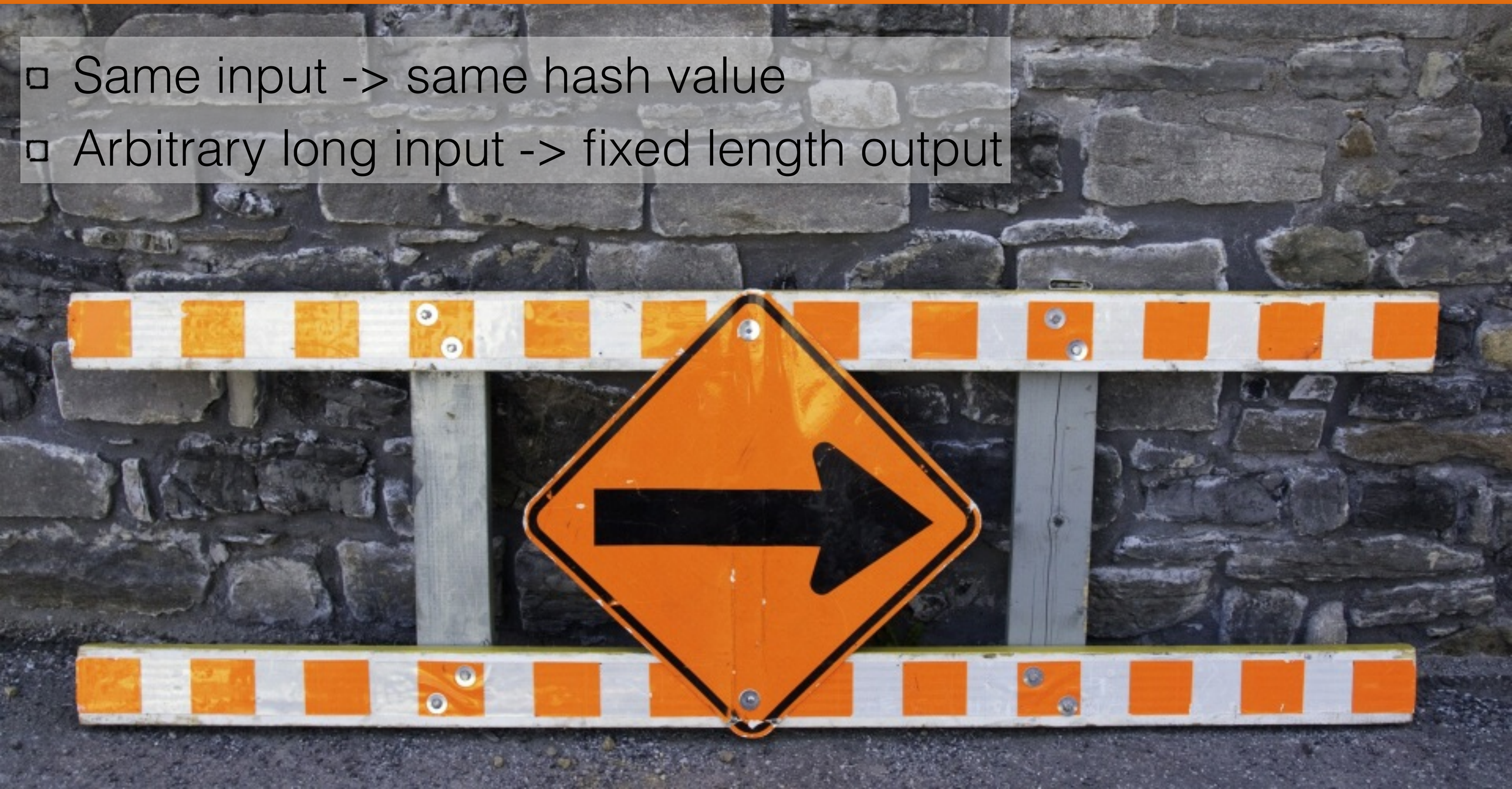
Demo

Hashes

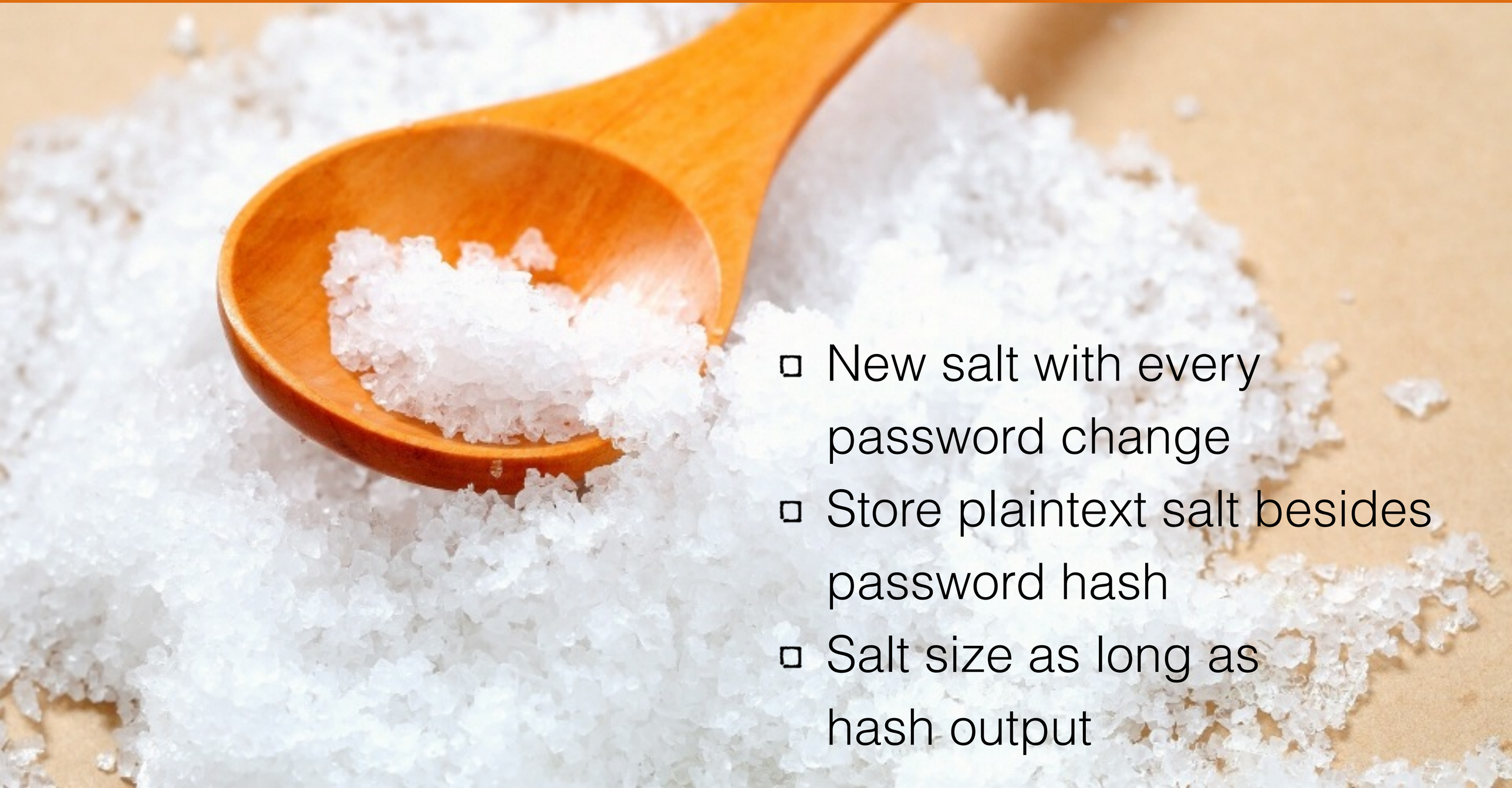
xato.net/passwords/more-top-worst-passwords

Hashing is a fast one way operation

- Same input -> same hash value
- Arbitrary long input -> fixed length output



A hashed password without an individual salt is neither safe nor unique



- New salt with every password change
- Store plaintext salt besides password hash
- Salt size as long as hash output

Special password hashing algorithms slow down brute force attacks

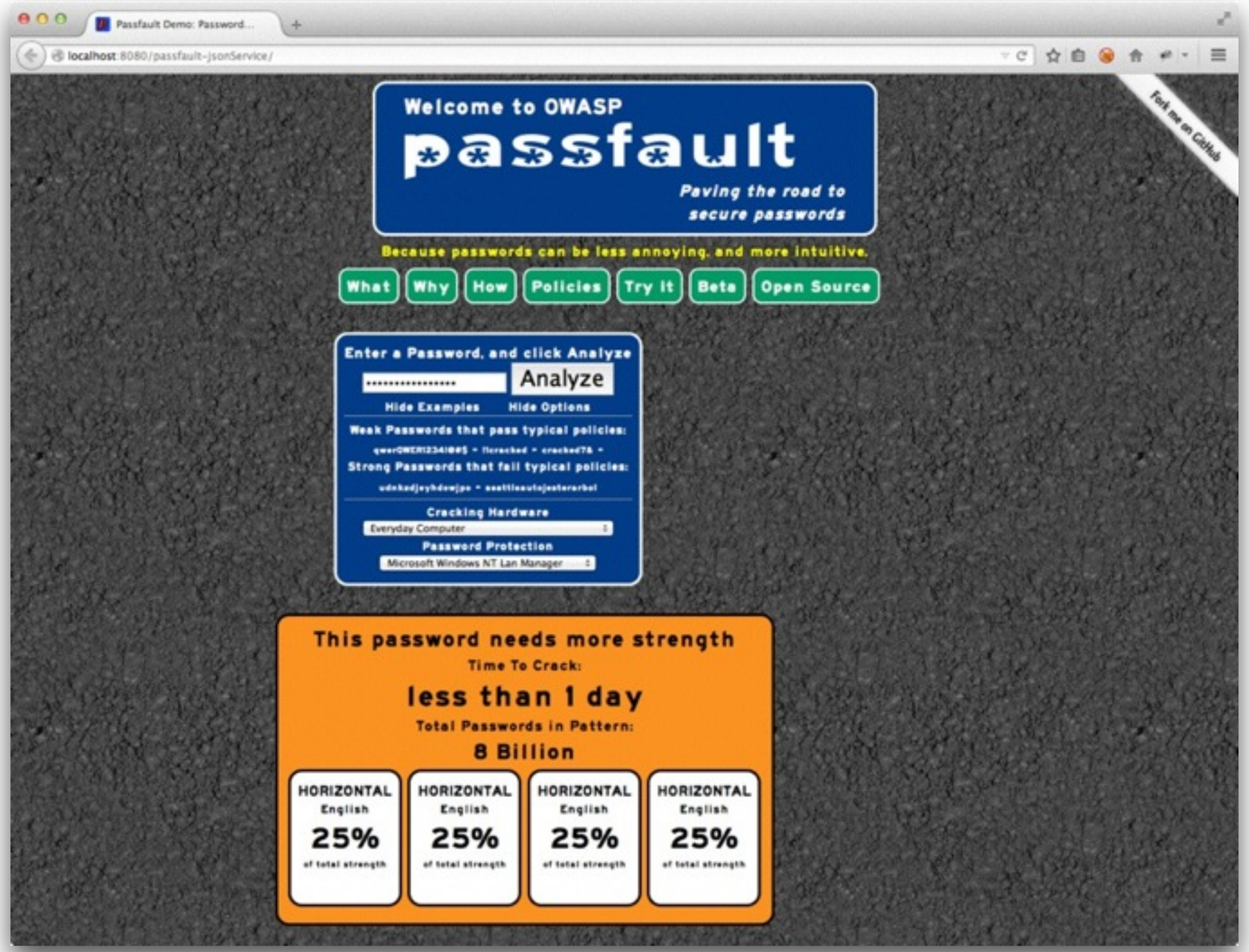
1. **PBKDF2** - multiple rounds
2. **scrypt** - resource intensive
3. **bcrypt** - iteration count

Apache Shiro provides safe defaults for encrypting and hashing

- Multiple iterations configurable
- Built-in salt generation
  - Base salt (private)
  - Public salt

Demo

OWASP Passfault analyzes password strength



www.owasp.org/index.php/OWASP_Passfault

- Don’t invent crypto, use it
- Crypto libraries come with limitations but easier usage
- Change your keys frequently

Dominik Schadow

BridgingIT GmbH
Königstraße 42
70173 Stuttgart

dominik.schadow@bridging-it.de
www.bridging-it.de

Blog blog.dominikschadow.de
Twitter @dschadow

Demo Projects
github.com/dschadow/JavaSecurity

Cryptographic Storage Cheat Sheet
www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

CrypTool
www.cryptool.org

Pictures
www.dreamstime.com